1 Forelesning 10

1.1 Graf

Grafer er en datastruktur som består av

- Noder (Vertices)
- Kanter (Edges) - par av noder

Binære trær er grafer.

1.2 Hva kan grafer representere?

Noder	**Kanter**
Byer	Veier
Person	Vennskap
Datamaskin	Nettverkskabel
Spill posisjon	Gyldig trekk
Student, emnet	Tar emnet
Robot, Job	Assigned
Lag	Kamp

1.2.1 Gradtall

Antall kanet som inneholder noden

1.2.2 Naboskap

Alle noder som kan nås med en kant

$$N(b) = \{a, c, f\} \wedge N[b] = \{a, b, c, f\}$$

1.2.3 Sykel

Noder og kanter som gjør at du kan gå fra en node og tilbake til seg selv mens du bare besøker hver node i sykelen akkurat en gang.

1.2.4 Tre

Et tre er en sammenhengende graf uten sykler.

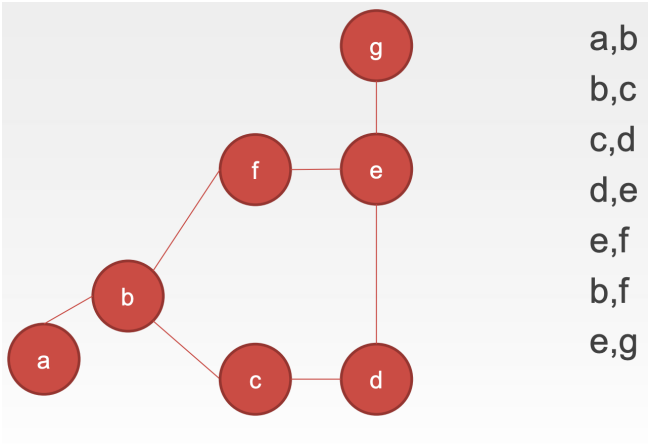
1.3 Graf datastruktur

- Viktige metoder i `Graph<V,E>`
  - `boolean adjacent (V a, V b)`
  - `Iterable<V> vertices()`
  - `Iterable<E> edges()`
  - `List<V> neighbours(V v)`
  - `add/remove metoder?`

1.3.1 Kjøretid i grafer

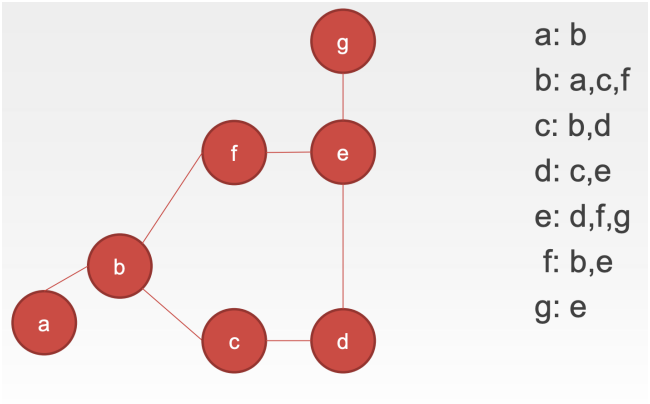
- Kjøretid i grafer avhenger av 2 parametere
  - N er antall noder
  - M er antall kanter
- Noen ganger er man mer presis
  - D = degree(v)
  - L = length of longest cycle
  - C = number of components

1.3.2 Graf datastruktur - Kantlise



Metode	**Kjøretid**
Ajacent	$O(m)$
Vertices	$O(n)$
Edges	$O(m)$
Neighbours	$O(m)$
addVertex	$O(1)$
addEdge	$O(1)$

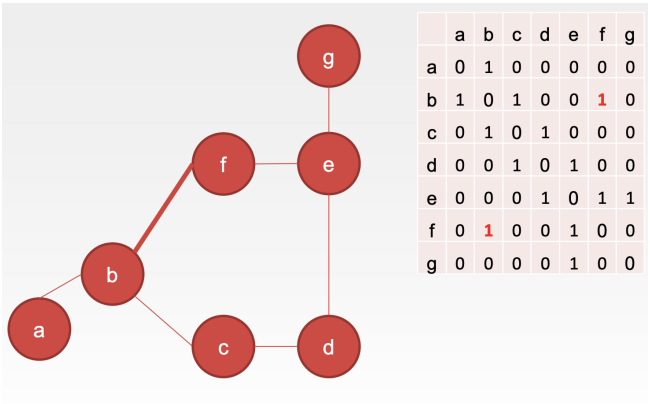
1.3.3 Graf datastruktur - Naboliste



- For hver node, hold liste av naboer
- Hold en liste av noder `List<V>`

Metode	Kjøretid
Adjacent	O (degree)
Vertices	$O(n)$
Edges	$O(m)$
Neighbours	O (degree)
addVertex	$O(1)$
addEdge	$O(1)$

1.3.4 Graf datastruktur - Nabomatrise



- 2 dimensjonell boolean tabell
- Bruker mye minne

Metode	**Kjøretid**
Adjacent	$O(1)$
Vertices	$O(n)$
Edges	$O(n^2)$
Neighbours	$O(n)$
addVertex	$O(n^2)$ or $O(n)$
addEdge	$O(1)$

1.4 Sammenhengende komponenter

Hvordan kan vi finne alle sammenhengende komponenter? Vi skal nå se på en algoritme som finner sammenhengende komponenter i en graf.

1. Søke i grafen med noe vi kaller bredde først søk.
2. Søke i grafen med noe vi kaller dybde først søk.
3. Bruke Union-Find (vil lære om dette snart)